WHAT IS CLAIMED IS:

1. A frame handler for application-level memory management, the frame handler comprising:

an associated block of memory divided into instances such that data elements may be stored in the instances;

a data structure identifying the unused instances within the block of memory; and

an application interface operable to receive a request for an unused instance from a software application,

wherein the frame handler is operable to identify an unused instance in response to a request received by the application interface.

2. The frame handler of claim 1 wherein the associated block of memory is divided into frames.

3. The frame handler of claim 2 wherein each frame is divided into instances.

4. The frame handler of claim 2 wherein the data structure includes a tree.

5. The frame handler of claim 4 wherein the tree is an AVL tree.

6. The frame handler of claim 4 wherein the tree includes a node associated with each frame.

7. The frame handler of claim 6 wherein each node is associated with a list of unused instances within the associated frame.

8. The frame handler of claim 7 wherein the list of unused instances is represented as a ring structure.

9. The frame handler of claim 6 further comprising an anchor including:

an empty list storing each node having no unused instances; and

a non-empty list storing each node having unused instances.

10. The frame handler of claim 1 further comprising an operating system interface operable to allocate a block of memory such that the frame handler is operable to allocate an additional block of memory when the block of memory is exhausted.

11. A method for allocating memory in a computer system, the method comprising:

outputting a request from an application to an operating system for allocation of a block of memory by the operating system to the application;

accessing the block of memory at the application;

dividing the block of memory into frames;

dividing each of the frames into instances, with each instance operable to store data and associated with an application-defined instance type; and

maintaining a data structure indicating each unused instance.

12. The method of claim 11 wherein maintaining a data structure indicating each unused instance includes creating a node corresponding to each of the frames.

13. The method of claim 12 wherein maintaining a data structure indicating each unused instance further includes associating a list of unused instances with each node.

14. The method of claim 13 wherein associating a list of unused instances with each node includes creating a ring data structure comprised of unused instances.

15. The method of claim 12 wherein maintaining a data structure indicating each unused instance further includes organizing the nodes in a tree structure.

16. The method of claim 15 wherein the tree structure is an AVL tree.

17. The method of claim 12 further comprising creating an anchor data structure including a ring including an empty list and a non-empty list.

18. The method of claim 17 wherein maintaining a data structure indicating each unused instance further includes placing nodes with unused instances in the non-empty list and placing nodes without unused instances in the empty list.

19. The method of claim 12 wherein dividing the block of memory into frames includes associating a frame identifier with each of the frames.

20. The method of claim 19 wherein each node includes the frame identifier of its associated frame.

21. A method comprising:

assigning a first identifier that is associated with a first memory portion to a first node;

linking a first list of instances to the first node, the first list of instances corresponding to divisions of the first memory portion;

assigning a second identifier that is associated with a second memory portion to a second node;

linking a second list of instances to the second node, the second list of instances corresponding to divisions of the second memory portion;

constructing a data structure using a plurality of nodes including the first node and the second node; and

selecting available instances from the instances for data storage by an application, wherein the instances are associated with an application-determined instance type.

22. The method of claim 21 wherein constructing a data structure comprises constructing an AVL tree using the plurality of nodes.

23. The method of claim 22 wherein selecting available instances comprises traversing the data structure to locate the available instances.

24. The method of claim 22 further comprising superposing a linear list over the data structure, wherein the linear list includes a first pointer to an empty subset of the plurality of nodes that has no associated memory available for use by the application and a second pointer to a not_empty subset that has associated memory available for use by the application.

25. The method of claim 24 wherein the first node is a first not_empty node in the not_empty subset, and selecting available instances comprises:

following the second pointer to the first node; and

using the first list of instances as the available instances.

26. The method of claim 25 further comprising:

re-setting the second pointer to a second not_empty node in the not_empty subset, and

including the first node in the empty subset.

27. The method of claim 21 further comprising:

determining an origin list from which the available instances were selected; and

returning the available instances to the origin list.

28. The method of claim 27 wherein determining the origin list comprises matching an identifier of the available instances to the first identifier or the second identifier.

29. The method of claim 28 wherein matching the identifier comprises following a pointer to a first not_empty node of a not_empty subset of the plurality of nodes, the not_empty subset including not_empty nodes with associated memory available for use by the application.

30. The method of claim 21 wherein the first memory portion includes a frame into which a block of memory allocated from the operating system is divided.